HLSUA
FORUM XLIII
Proceedings

PHOENIX CIVIC PLAZA
PHOENIX, ARIZONA
OCTOBER 5-8, 1986

The MSQ Database System

I. J. Davis

Department of Physics and Computing
Wilfrid Laurier University
Waterloo, Ontario, Canada

ABSTRACT

MSQ is a database system, developed by Wilfrid Laurier University, that supports a wide variety of different applications. MSQ runs under the CP-6 operating system, and combines relational, hierarchical, and network concepts into an integrated database management system. Records in an MSQ database are organised into user defined fields, that may be dynamically created, deleted and modified. Each field in an MSQ record contains an arbitrary number of subfields.

MSQ supports a powerful (but natural) query language, that allows records to be efficiently retrieved using multi-key hashing. This query language allows users to navigate through data in one or more MSQ databases, by supporting a variant of the classical relational join, and allows fields to be derived and reported in numerous ways. MSQ supports a large variety of reporting methods, allowing records to be sorted and reported in a horizontal, vertical, screen, or many-up label format.

An MSQ database may be accessed and updated concurrently by many users, either directly, or indirectly via scripts written in MSQ's query language. These scripts support input, output, command execution, and conditional branching, allowing flexible menu driven systems to be easily implemented. This paper describes these and other features of MSQ.

August 17, 1986

The MSQ Database System

I. J. Davis

Department of Physics and Computing
Wilfrid Laurier University
Waterloo, Ontario, Canada

1. History:

In the spring of 1984 Wilfrid Laurier University began migrating its computing facilities from CP5 to CP6. At that time several departments were using the VIP [1] database system, which could not easily be converted to run under CP6. As a result Wilfrid Laurier University decided to develop MSQ. Initially MSQ operated as an electronic card file system with support for records containing user defined fields. Each of these fields contained an arbitrary number of subfields, each containing arbitrary data. MSQ provided facilities to create and selectively update these records, the ability to retrieve records satisfying specified selection criteria, and allowed the user to report the contents of these (optionally sorted) records in a wide variety of formats.

Following MSQ's initial release, Xerox agreed to purchase certain rights to MSQ, if MSQ could be upgraded to meet Xerox's requirements. These requirements included implementing the necessary locking so that databases could be safely accessed by concurrent users, and extending the number of records that MSQ supported within any one database from 10,000 to 1,000,000. These and other modifications were done in early 1985.

In the summer of 1985 Wilfrid Laurier University began investigating how it might consolidate all of its database requirements under a single database product. Because of this investigation various limitations of MSQ became apparent, and effort was spent rectifying these. The most extensive enhancement to MSQ involved implementing a variant of the classical relational join [2]. This enhancement allows individual MSQ databases to be used either as electronic card files, or as components of a larger network of record types.

2. Fields

A user creates a new field within an MSQ database by merely defining it. Each field has numerous optional attributes, which control how data may be entered into the newly defined field, accessed, and reported. Fields once defined can be safely modified, renamed, or deleted, since the field definition is essentially independent of the data stored within that definition. The data entered into a field may be single or multi-valued, optional or mandatory, and may be restricted to being either numeric or a valid date. The maximum number of decimal places associated with numeric data is also maintained as part of the field definition. Each field is also assigned an access level, that controls who may retrieve and update the field.

In addition to fields that contain user data, fields may be defined that contain system generated time-stamps. These time-stamps may be accessed but not updated by the user.

A variety of functions exist that allow users to define new virtual fields from existing fields. Values in these existing fields may be combined, subdivided, or extracted in various ways, and may also be totalled, counted or used in complex numeric expressions. These resulting derived fields may themselves be used in defining other fields.

MSQ also supports joined fields. Such fields are completely defined by their parameters. These parameters consist of a field name whose values are used as search keys, a field in an arbitrary MSQ database whose values are to be matched against this key field, and a field (or joined field) from this same arbitrary database whose values are to be retrieved whenever a match occurs.

3. Example

Consider a student database that uses the following fields :

Field	Description
Id	Single valued positive numeric value identifying student.
Full-name	This is derived from the fields *Prefix*, *First-name*, *Last-name*
Spouse	The identifier of this student's spouse.
Friends	The identifier of this student's known friends.
Highschool	The institute that the student attended.

This database, and a related institute database contain:

Sample data in student database

Id	Full Name			Spouse	Friends	Highschool
1	Mr	Joe	Little	2	3	C.H
2	Mrs	Jane	Little	1	3	C.T
3	Miss	Mary	Small		1	C.H
					2	

Sample data in institute database

Id	Name	Address
C.H	Collegiate High School	41 Underhill Road Andover Ontario
C.T	Central Technical School	39 Uppermill Drive Norunder Manitoba

Suppose that the record pertaining to "Mr. Joe Little" has been selected from the student database. Then the joined fields listed below, return the indicated values.

Join Field	Data returned
spouse→id.full-name	Mrs. Jane Little
spouse→id.friends→id.last-name	Small
highschool→institute:id.name	Collegiate High School
highschool→highschool.full-name	Mr. Joe Little
	Miss. Mary Small

These values appear to reside in the record pertaining to "Mr. Joe Little". They can be used to assist in sorting and the production of any report. The only distinction that MSQ makes between a genuine field within a record, and a derived field, such as one produced by using a relational join, is that the latter may not be directly updated. Thus, for example, the derived field *full-name* that is produced by the function $CONCAT(*prefix* '. ' *first_name* ' ' *last_name*) cannot be directly updated. Instead users must modify *prefix*, *first-name*, and *last-name* appropriately.

4. Record selection

MSQ provides a user friendly syntax that allows users to select records based on the contents of specified fields within these records. The value of fields can be compared against given values for equality, inequality, and range.[1] The various criteria involved in such selection are combined using "and", "or", and "not" in the normal manner. Brackets may also be used. The resulting relational expression may be used either as part of an MSQ command that operates on records, or may be used to specifically select records, omit records, keep records from a previously selected set, or include new records in a previously selected set.

Tests for equality (and inequality) may freely use wildcarding. A "?" appearing anywhere within a word is interpreted as zero or more characters within that word; otherwise, it is interpreted as zero or more words. Additional "?"s immediately following a "?" may be used if necessary to indicate the exact number of characters or words being wildcarded. Less obviously, MSQ also treats blanks as wildcards that match any sequence of punctuation symbols, avoiding many of the problems that punctuation might otherwise cause.

In addition to allowing relationships that depend on the contents of a field, MSQ also allows relationships that depend on the number of values occurring in a field, or the size of these values. MSQ also allows relationships to be restricted to specified subfields within fields, or to unspecified subfields occurring at the same subfield offset within different fields.

[1] MSQ is not case sensitive unless specifically requested to be by the user.

5. Access methods

Each MSQ record is assigned a unique numeric key which provides direct access to that record. This key is stored as a value in a special field, allowing users to access records rapidly by key, while conforming to the standard selection syntax. When users do not explicitly provide the record keys, MSQ must itself determine these values. To assist in this process users may associate secondary indices with fields. Two distinct types of secondary index are supported, both of which utilise hashing rather than inverted indices. A third access method that does use indexing is currently being developed. These secondary indices facilitate rapid retrieval of data, even when wildcarding is used.

The first type of hashing that can be applied to a field examines each word of data stored in this field and, based on the start character and length of each word, groups the record containing this word, with other records containing words in this field with the same start character and length.

This type of hashing is very effective when retrieving text containing many words, or when retrieving text distinguishable by the first character. However, it is of no use when all values start with the same character, and are of the same length. Consider for example a field that contains recent dates. All start with 19?? and are the same length. To overcome this limitation users may alternatively elect to hash by a specified number of character positions rather than just the first. This type of hashing gives very much faster retrieval when fields contain only a few words, but becomes inefficient when fields contain many words.

When one or more words within such a hashed field are required, MSQ first establishes the superset of records possibly containing the desired data, and then eliminates false drops by serially scanning the preselected records. For example, if *book-title* is hashed by first character and length, and *published* is hashed on the first three characters, then the query *select when book-title is like 'WOM??N IN ACT?' and published 1985*, results in MSQ using its secondary indices to immediately determine the set of records containing a *book-title* with a five letter word starting with 'W', a two letter word starting with 'I', and a three or more letter word starting with 'A'. Similarly, the sets of records containing *published* beginning with '1', continuing with '9', and containing an '8' in the third position of some word are also each established. Finally, the boolean 'and' of all these sets is determined, resulting in a very small set of potentially valid records. These remaining records are then scanned serially to eliminate such possible titles as 'ACTIONS OF WOMEN IN LOVE' published in 1983.

Secondary indices are also used when relational joins are performed. When a relational join is used to retrieve data, MSQ internally generates the query (or queries) needed to obtain the records containing the desired data. Thus *list friends→ id.full-name* causes MSQ to obtain the list of friends for each record selected, then to request those records containing an identifier matching any one of these friends, and then to retrieve the desired names of these friends. When joined fields are used as part of a query, MSQ still uses secondary indices but in a slightly different manner. For example, suppose that the command *Select friends→ id.last-name like Joe* is issued. MSQ first establishes the identifier of each record containing the name Joe. It then selects all records having a value in the friends field matching any one of these identifiers.

The records associated with each hash set are represented internally by a fixed length bitmap[2] and stored in a relative file, allowing rapid retrieval. Users have a great deal of

[2] MSQ automatically extends the size of bitmaps whenever necessary.

control over the number of such bitmaps stored and thus the storage space required by these secondary indices. Commands within MSQ allow the statistical distribution of records in these hashed sets to be reported, and to be fine tuned, if desired. Commands are also available that allow the values in these bitmaps to be verified following system crashes, and corrected when erroneous.

6. Reports

MSQ provides a variety of reporting facilities and options, allowing numeric values and dates to be displayed in over one hundred styles, with support for currency, ten digit precision, hours, minutes, etc.

For an easily read report, the user can select two types of horizontal listing. For business reports, data can be output in a columnar format with user defined titles, headings, control breaks, and totalling. Output that exceeds the column width can be truncated, or wrapped across lines on either character or word boundaries. For interactive work users can request screens containing both user specified text and data. MSQ automatically creates multiple screens across the page, if the output platen width is large enough, allowing many-up labels to be produced.

Records can be sorted in ascending or descending sequence using up to ten fields as sort keys. Sorting may be based on specific subfields within each field, or may result in a "sort explosion" when the fields used contain multiple subfields. By default sorting is not case sensitive, and may optionally be applied to fields from which all punctuation has been removed.

Various additional reports can be produced that provide statistics about the data retrieved. The frequency with which values occur in a field or collection of fields can be reported in a variety of ways, as can the widths of these values. Statistics can also be obtained about values occurring in specified fields. These statistics include maximum, minimum, average values, as well as standard deviation of values etc. All statistics are calculated using twenty digit precision.

7. Update

Records are generally examined interactively and modified using standard line editing facilities, subfield by subfield. The selected records may be presented for update in any sorted sequence, as may the fields or subfields that the user wishes to update. Fields within each record may either be modified serially, or in parallel. New subfields can be inserted at any point within a sequence of subfields, and old subfields joined or divided.

Each subfield is verified at the time of entry and rejected if incorrect. Verification ensures that a subfield conforms to its field definition, and may optionally constrain its size. If desired, the data entered may be used to lookup and display arbitrary joined fields, with successful lookups optionally causing the update to be either accepted or rejected.

MSQ allows numeric data to be entered in a variety of formats and allows dates/times to be entered in either numeric format or with the month being entered as a standard three digit abbreviation. Internally dates are stored as numeric values, which are themselves always stored as binary words, allowing ten digit accuracy. As each record is updated, MSQ automatically updates all secondary indices and time-stamps.

Users occasionally wish to restructure databases so that derived fields are accessed via different paths, or stored in different ways. This can be accomplished easily, since any genuine field can be assigned the values present in other genuine or derived fields. This facility also allows conversion and rescaling of numeric data within any one field.

Because a certain amount of data exchange occurs between MSQ databases, MSQ provides facilities to dump selected records into a serial file, and subsequently restore the contents of these records into arbitrary fields in any MSQ database. These facilities are also useful when reorganising MSQ databases. Any field in the dumped record may be used to define the key of the record receiving this dumped data. Alternatively, MSQ may be instructed to generate new records when the restore occurs. When the data being restored updates an existing record, options exist that allow this existing record to be deleted, to be left unchanged, or to be merged with the data being restored.

MSQ interrupts its current activities whenever the break key is pressed. If MSQ is in the process of updating data it ensures that the data updated is internally consistent before acknowledging the break. It then informs the user of the number of records updated, and the last record updated, so that the user can continue later with updates, if required.

8. Synonyms

MSQ allows fields to be assigned an arbitrary number of names and allows any fragment of an MSQ command to be assigned a synonym. By assigning synonyms to a command, lengthy commands can be readily invoked. This is particularly important when generating complex reports or screens where a single command can span many lines. Synonyms once defined can be edited, renamed, and deleted.

Occasionally a whole sequence of commands is required to be stored and executed as a single unit. MSQ allows such commands to be stored in external files and executed as a unit on request. In addition MSQ automatically searches for specially named files of commands when first invoked, so that any desired initial command sequence can be executed transparently.

Users often wish to implement a superstructure of user commands, on top of the natural language facilities provided by MSQ. Typically, administrators wish to develop a menu driven system applicable to their application, so that the knowledge needed to access their system is minimal. To support such requirements MSQ provides the ability to output arbitrary text, and to prompt for input that is subsequently embedded in arbitrary commands. In addition MSQ provides a simple interpretive language that allows transfer of control to be based on input, or other parameters. Transfer of control can also be specifically defined by the user when breaks occur.

A further interface to MSQ is currently being developed, that will allow programs written in a variety of conventional languages to interact with MSQ. Such a facility is needed to support non-standard update and reporting requirements that may arise in complex database systems. Having developed programs that support non-standard reporting or update requirements, these programs will be invoked from within MSQ by using special commands, thus integrating their functionality with MSQ's. Some of these programs may alternatively be invoked transparently by MSQ, when verifying or formatting non-standard data fields.

9. Security

MSQ provides various levels of security. CP6 allows files to be assigned access rights, that restricts their unauthorised use. This facility has been used to allow selective sharing of MSQ databases, and to assign certain users read only access to MSQ data.

Since the users identification is available as a preset parameter within MSQ, administrators who are developing a menu driven database system can easily control the access to their database that is to be allowed by each user. They can if desired prompt for additional information allowing the identification of distinct individuals sharing common accounts to be determined. If users are allowed no access to general CP6 facilities, MSQ can be instructed to perform a line-disconnect following any attempt to exit from MSQ.

Internally, each field in an MSQ database is assigned an access level of between 0 and 9. All users are initially assigned an access level of 9 when they first invoke MSQ, but can change their access level by providing the appropriate password. MSQ will not allow users to modify fields having an access level smaller than their current access level. MSQ will allow fields to be read providing that the field being read has an access level which is no more than one smaller than the users current access level.

Synonyms can only be changed by users whose access level is 2 or smaller. Other components of the MSQ data dictionary can be modified only by users at access level 1 or 0. All passwords are accessible and modifiable only by a user at access level 0.

The procedure that allows users of one MSQ database to access another is carefully controlled to avoid security violations. A level 0 user can give a foreign database read and update access to their database. The foreign database is also provided with some smallest access level, that it can assume when accessing this database. Any modification to such access rights are made by this level 0 user.

Records are locked against concurrent writers internally, to avoid update inconsistencies. Global changes to either the data dictionary, or to the secondary indices require that the MSQ database be transparently opened for exclusive access. No locks are placed on records being read, since records are updated atomically.

One further level of security exists, which endevours to protect both users and data against software failure. Numerous checks are constantly performed to verify that records read conform to the anticipated MSQ format; that the secondary indices contain correct data whenever these are updated; that software is behaving as expected, etc. Collectively, these silent checks have helped to eliminate many bugs.

10. Technical characteristics

MSQ is a shared processor, that is currently divided into 6 overlays. Upon being loaded it occupies approximately 42k bytes. However, MSQ makes extensive use of dynamic memory, and cannot be expected to run satisfactorily in less than 64k. Most users of MSQ run successfully in less than 128k of memory, but very large databases may potentially require more. MSQ provides a detailed account of memory usage, when requested.

The amount of disk space required to store an MSQ database, is dependent both on the amount of data contained in the data file, and on the number of secondary indices contained in the hash file. The space occupied by the data file tends to be less than expected, since conventional storage methods do not support variable length text. The hash file generally occupies much less space than the data file, unless all fields are heavily indexed, or large gaps exist in the record keys used.

MSQ is written entirely in PL/6, and currently consists of more than 19,000 lines of source code. It uses the parser "PARTRGE" [3], and standard sort facilities. It relies very heavily on almost all of the facilities provided by the CP6 operating system. MSQ is accompanied by a short users guide [4], and by a very much larger users reference manual [5].

11. Conclusions

Prior to the development of MSQ, Wilfrid Laurier was using VIP to support six different applications. Wilfrid Laurier University is currently using MSQ in more than seventy five different areas. Xerox is almost certainly using MSQ in many others. At least fifty users have elected to start using MSQ since its initial release in 1984, and many more routinely access MSQ databases. Generally, users have found MSQ easy to use, and effective in meeting their database requirements, even when these users have had little previous computer experience. Users have been encouraged to submit suggestions on how MSQ can be improved, and these suggestions have been implemented whenever possible. Even if this paper were to provide a detailed description of MSQ, it would necessarily be incomplete. MSQ is constantly being reviewed and enhanced.

Acknowledgements

The author would like to stress that many people contributed to the development of MSQ. Special thanks are extended to Ejvind Andersen for supervising the development of MSQ, and for assisting in the design and testing of this product. Thanks are also extended to Hart Bezner for funding much of the development of MSQ, and to Dave Mathews for continuing to motivate interest in MSQ and other database products available to the Wilfrid Laurier community. Finally, I wish to thank Dave Mathews and Carl Seger for many helpful comments relating to this paper.

References

1. The VIP (Versatile Information Processor) Reference Manual. Comshare, Toronto, Canada. 06014D. 1981.
2. J. D. Ullman. Principles of Database Systems. Computer Science Press. 1980. (379 pages).
3. The CP6 Monitor Services Reference Manual. Honeywell Information Systems. CE33-02. 1982.
4. E. Andersen. The MSQ Quick Reference Guide. Wilfrid Laurier University. 1985. (33 pages).
5. E. Andersen. The MSQ information management system, users guide. Wilfrid Laurier University. 1986. (100+ pages).